Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (currently amended) A queue comprising:

a first ~~queuing area~~ queue configured to enqueue and dequeue data units, the first ~~queuing area~~ queue including a plurality of parallel sub-queues that queue a plurality of parallel data units;

a second ~~queuing area~~ queue configured to receive data units from the first ~~queuing area~~ queue when the first ~~queuing area~~ queue has data units available to be dequeued, the second ~~queuing area~~ queue including a first buffer configured to store a first set of the parallel data units and a second buffer configured to store a second set of the parallel data units; and

bypass logic coupled to the second ~~queuing area~~ queue, the bypass logic configured to bypass the first ~~queuing area~~ queue and to forward data units to the second ~~queuing area~~ queue when the second queue ~~queuing area~~ is ready to receive data units and the first ~~queuing area~~ queue is empty.

2. (currently amended) The queue of claim 1, wherein the data units consists of memory access requests.

3. (canceled)

4. (canceled)

5. (currently amended) The queue of claim 1, further comprising:

an encoding component coupled to the bypass logic and the first and second buffers, the encoding component configured to read data units from the first and second buffers, wherein the encoding component gives data units in the first buffer higher priority than data units in the second buffer.

6. (cancel)

7. (currently amended) The queue of claim 1, further comprising:

masking logic coupled to the output of the first and second buffers, the masking logic configured to restore data units to ~~requests of the set of arbitration requests of~~ the first and second buffer that were not ~~read~~ output from the first and second buffers.

8. (currently amended) A method of masking latency in a device ~~queue~~, the method comprising:

receiving incoming data items for [[the]] a queue that include a plurality of data items that are input to the queue for each cycle of the queue;

forwarding the incoming data items to a buffer when the queue is empty and the buffer is free to receive data items, wherein the buffer includes a first buffer and a second buffer, and wherein higher priority data items are stored in the first buffer and lower priority data items are stored in the second buffer;

enqueuing the incoming data items in the queue when the queue contains data items or the buffer is not free to receive data items;

dequeuing data items from the queue to the buffer when the buffer is free to receive data items; and

transmitting the data items from the buffer as the output of the device ~~queue~~.

9. (original) The method of claim 8, wherein the data items are memory access requests.

10. (canceled)

11. (canceled)

12. (currently amended) A method of masking latency in a queuing device ~~queue~~, the method comprising:

receiving incoming data items for [[the]] a queue;

forwarding the incoming data items to a buffer when the queue is empty and the buffer is free to receive data items, the buffer including a first buffer and a second buffer, and wherein higher priority data items are stored in the first buffer and lower priority data items are stored in the second buffer;

enqueuing the incoming data items in the queue when the queue contains data items or the buffer is not free to receive data items;

dequeuing data items from the queue to the buffer when the buffer is free to receive data items; and

transmitting the data items from the buffer as the output of the queuing device ~~queue~~,

wherein the data items in the second buffer are moved to the first buffer when the first buffer is free to receive data items.

13. (currently amended) The method of claim 8, wherein two data items are transferred from the first and second buffer per cycle as the output of the ~~queue~~ queuing device whenever the first and second buffer contain at least two data items.

14. (original) A network device comprising:

a request manager configured to receive memory requests;

a plurality of parallel processors configured to receive the memory requests from the request manager; and

a memory request arbiter configured to receive the memory requests from the plurality of processors, the memory request arbiter transmitting the memory requests to a memory system based on an arbitration scheme, the memory request arbiter including:

an input port connected to receive the memory requests from the plurality of processors,

a queue corresponding to each of the plurality of parallel processors, each of the queues configured to enqueue and dequeue memory requests of the corresponding parallel processor, and

a buffer configured to receive memory requests dequeued from the queues when the queues contain memory requests and to receive memory requests directly from the input port when the queues do not contain memory requests.

15. (original) The network device of claim 14, wherein the buffer further comprises:

a first buffer configured to store a first set of parallel memory requests; and

a second buffer configured to store a second set of parallel memory requests.

16. (original) The network device of claim 15, wherein the memory request arbiter further includes:

bypass logic coupled to the buffer and the queues, the bypass logic causing the received memory requests to bypass the queues and to be received directly by the buffer.

17. (original) The network device of claim 16, further comprising:

an encoding component coupled to the bypass logic and the first and second buffer, the encoding component reading memory requests from the first and second buffer, wherein the encoding component gives memory requests in the first buffer higher priority than memory requests in the second buffer.

18. (original) The network device of claim 17, wherein the encoding component reads multiple memory requests per clock cycle from the first and second buffers.

19. (original) The network device of claim 14, wherein the network device is a network router.

20. (currently amended) A device comprising:

means for receiving incoming data that includes a plurality of data for each cycle;

means for buffering the data before transmitting the data in a first buffer and a second buffer, in which higher priority data is stored in the first buffer and lower priority data is stored in the second buffer;

queue means;

means for forwarding the received incoming data to the means for buffering when the queue means is empty and the means for buffering is free to receive data;

means for enqueuing a plurality of the incoming data to the queue means, in a cycle of the queue means, when the queue means contains data or the means for buffering is not free to receive data; and

means for dequeuing data from the queue means to the means for buffering when the means for buffering is free to receive data.

21. (previously presented) The device of claim 20, wherein the data consist of memory requests.

22. (original) An arbiter comprising:

a queue configured to enqueue data items at a first stage of a plurality of stages and dequeue the data items at a last stage of the plurality of stages of the queue;

a multiplexer having a plurality of inputs connected to different stages of the queue, the multiplexer outputting selected ones of the data items read from the queue; and

arbitration logic coupled to the queue, the arbitration logic controlling the multiplexer to output the selected ones of the data items by selecting a predetermined number of data items from the queue during an arbitration cycle, the arbitration logic giving higher priority to data items in later stages of the queue.

23. (original) The arbiter of claim 22, further comprising:

bypass logic coupled to the queue, the bypass logic causing the data items to bypass the queue and to forward the data items to the multiplexer when the queue is empty.

24. (previously presented) The arbiter of claim 22, wherein the data items consist of memory access requests.

25. (original) The arbiter of claim 22, wherein the queue includes a first-in-first-out (FIFO) queue.

26. (new) The queue of claim 1, wherein the second queue is further configured to:

output up to a predetermined number of the first set of the parallel data units in a clock cycle and, when a number of data units in the first set of parallel data units is less than the predetermined number, output one or more of the second set of parallel data units in parallel with the first set of the parallel data units.

27. (new) The method of claim 8, wherein transmitting the data items from the buffer includes:

transmitting up to a predetermined number of data items from the first buffer in a first cycle of the queue and, when the first buffer does not include the predetermined number of data items, transmitting additional data items from the second buffer, up to the predetermined number of data items, in the first cycle of the queue.